**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный технический университет»**

**(ФГБОУ ВО «ВГТУ», ВГТУ)**

**Факультет** экономики, менеджмента и информационных технологий

**Кафедра** систем управления и информационных технологий в строительстве

**Курсовой проект**

по дисциплине: «Программирование мобильных приложений»

на тему: «Разработка программного обеспечения демонстрирующие работу молочной фермы»

Выполнил студент: Житенев С.В.

Группа: бИС - 181

Руководитель: доцент, к.т.н. Черменев Д.А.

Работа защищена « » 2022 г.

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Воронеж 2022

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc92641236)

[1 Анализ предметной области 4](#_Toc92641237)

[2 Анализ средств разработки 7](#_Toc92641238)

[2.1 Многопоточность. Синхронизация потоков 7](#_Toc92641239)

[2.2 Рефлексия. Делегаты и события 8](#_Toc92641240)

[3 Разработка программного обеспечения 10](#_Toc92641241)

[3.1 Выбор средств реализации 10](#_Toc92641242)

[3.2 Пример работы программного обеспечения 11](#_Toc92641243)

[ЗАКЛЮЧЕНИЕ 15](#_Toc92641244)

[Список используемых источников. 16](#_Toc92641245)

[Приложение 18](#_Toc92641246)

ВВЕДЕНИЕ

Сегодня информационные технологии, которые улучшают условия жизни людей, стремительно развиваются. Значительно повышается уровень знаний, в связи с этим становится необходимым эффективно организовывать, сохранять и управлять доступом к ним. В современном мире просто невозможно представить жизнь без информационных технологий. Информационные технологии представляют весь накопленный опыт человечества в форматизированном виде, пригодном для прикладного использования. Вопрос о потребности в автоматизации является актуальным в наше время, в решении которого, несомненно, помогает программирование. Одной из основных задач можно рассматривать проблему ведения отчетности, а также оперативную корректировку данных при возникновении необходимости в этом. Для работы с данными используются системы управления базами данных.

Целью данного курсового проекта является разработка программного обеспечения демонстрирующего работу молочной фермы. Смоделировать работу молочной фермы. Реализовать классы – Механик, Склад, интерфейс – погрузчик. События – Склад заполнен – приезжает погрузчик и увозит молоко, сломалось оборудование (с некоторой долей вероятности) – механик чинит всё.

Программа разработанна в среде MS Visual Studio 2021.

1. Анализ предметной области

Темой данного курсового проекта является разработка программного обеспечения демонстрирующие работу молочной фермы.

Молочное хозяйство (молочная ферма) — агропромышленное предприятие, производящее молоко-сырьё, в основном от молочных коров, а также от коз, овец, лошадей, верблюдов и некоторых других сельскохозяйственных животных.

Крупный рогатый скот — основной вид животных в молочных хозяйствах: коровы дают более 500 млн тонн всемирных годовых надоев (2014), буйволицы — около 100 млн тонн (в основном — в Индии); годовое мировое производство козьего молока — 18 млн тонн, овечьего — 10 млн тонн. Хозяйства, как правило, содержат животных в течение всего их биологического цикла, устройство предприятий обычно следует за необходимостью раздельного содержания и ухода за животными разных возрастных групп. Разнообразие технологий содержания и обслуживания варьируется в зависимости от размеров хозяйств, доступных средств механизации и автоматизации, а также природно-климатических условий; обслуживание включает такие операции, как доение, кормление, уборку навоза, зооветеринарный сервис. Выработанное молоко на современных молочных хозяйствах подвергается первичной переработке, после чего отправляется специальным транспортом на предприятия молочной промышленности (молочные комбинаты, сырзаводы, маслозаводы, молочноконсервные заводы); некоторые предприятия выработанное сырьё или его часть перерабатывают на месте.

Постановка задачи:

В зависимости от задачи необходимо смоделировать ситуацию/процесс. В каждой модели есть набор возможных ситуаций. Для некоторых событий необходимо определить вероятность возникновения данного события. Интерфейс необходимо реализовать, используя 3 и более классов с использованием синхронизации потоков. На форме должно быть не менее 4 моделей. Ограничения накладываются на классы, которые реализуют интерфейсы. Для 4 моделей должно быть 2 объекта данных классов в сумме. При возникновении какого-то события 1 из объектов «лочится» и не доступен для использования в других моделях.

Для решения задач необходимо использовать:

1. Делегаты/события.

2. Многопоточность

3. Где необходимо рефлексию

4. Синхронизацию потоков

На форме должно быть динамическое изменение моделей – все должно двигаться. Иметь возможность добавлять несколько моделей на форму

1. Анализ средств разработки

2.1 Многопоточность. Синхронизация потоков

Одним из ключевых аспектов в современном программировании является многопоточность. Ключевым понятием при работе с многоопоточностью является поток. Поток предствляет некоторую часть кода программы. При выполнении программы каждому потоку выделяется определенный квант времени. И при помощи многопоточности мы можем выделить в приложении несколько потоков, которые будут выполнять различные задачи одновременно. Если у нас, допустим, графическое приложение, которое посылает запрос к какому-нибудь серверу или считывает и обрабатывает огромный файл, то без многопоточности у нас бы блокировался графический интерфейс на время выполнения задачи. А благодаря потокам мы можем выделить отправку запроса или любую другую задачу, которая может долго обрабатываться, в отдельный поток. Поэтому, к примеру, клиент-серверные приложения (и не только они) практически не мыслимы без многопоточности.

Основной функционал для использования потоков в приложении сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток - класс Thread.Процесс — это исполнение программы. Операционная система использует процессы для разделения исполняемых приложений.

Поток — это основная единица, которой операционная система выделяет время процессора. Каждый поток имеет [приоритет планирования](https://docs.microsoft.com/ru-ru/dotnet/standard/threading/scheduling-threads) и набор структур, в которых система сохраняет контекст потока, когда выполнение потока приостановлено. Контекст потока содержит все сведения, позволяющие потоку безболезненно возобновить выполнение, в том числе набор регистров процессора и стек потока. Несколько потоков могут выполняться в контексте процесса. Все потоки процесса используют общий диапазон виртуальных адресов. Поток может исполнять любую часть программного кода, включая части, выполняемые в данный момент другим потоком.

Платформа .NET предоставляет множество готовых примитивов синхронизации и потокобезопасных коллекций. Если при разработке приложения нужно реализовать, например, потокобезопасный кэш или очередь запросов — обычно используются эти готовые решения, иногда сразу несколько. В отдельных случаях это приводит к проблемам с производительностью: долгим ожиданием на блокировках, избыточному потреблению памяти и долгим сборкам мусора.

Синхронизация потоков это процесс, который позволяет выполнять все параллельные потоки в программе синхронно. Синхронизация позволяет избежать ошибок согласованности памяти, вызванные из-за непоследовательного доступа к общей памяти.

При выполнении нескольких потоков они будут неизменно взаимодействовать друг с другом, чтобы синхронизировать свое выполнение. Существует несколько методов синхронизации потоков:

* Блоки взаимного исключения;
* Переменные состояния;
* Семафоры.

2.2 Рефлексия. Делегаты и события

В [информатике](https://wikipedia.tel/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) отражение или рефлексия ([холоним](https://wikipedia.tel/%D0%9C%D0%B5%D1%80%D0%BE%D0%BD%D0%B8%D0%BC_%D0%B8_%D1%85%D0%BE%D0%BB%D0%BE%D0%BD%D0%B8%D0%BC) [интроспекции](https://wikipedia.tel/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [англ.](https://wikipedia.tel/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) reflection) означает процесс, во время которого [программа](https://wikipedia.tel/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) может отслеживать и модифицировать собственную структуру и поведение во время выполнения. [Парадигма программирования](https://wikipedia.tel/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), положенная в основу отражения, называется рефлексивным программированием.

Во время выполнения программных инструкций (кода) компьютеры обрабатывают данные, что приводит к их изменению, при этом компьютеры не изменяют код. Однако, в большинстве современных компьютерных архитектур код хранится как данные, и в некоторых языках программирования реализована возможность обрабатывать собственный код как данные, что приводит к изменению уже самого кода во время его выполнения.

Событие представляет собой автоматическое уведомление о том, что произошло некоторое действие. События действуют по следующему принципу: объект, проявляющий интерес к событию, регистрирует обработчик этого события. Когда же событие происходит, вызываются все зарегистрированные обработчики этого события. Обработчики событий обычно представлены делегатами.

Делегаты представляют такие объекты, которые указывают на методы. То есть делегаты - это указатели на методы и с помощью делегатов мы можем вызвать данные методы.

Для объявления делегата используется ключевое слово delegate, после которого идет возвращаемый тип, название и параметры.

1. Разработка программного обеспечения

3.1 Выбор средств реализации

Как было сказано ранее, программный продукт разрабатывается на языке программирования высокого уровня C# с использованием интерфейс программирования приложений Windows Forms. Рассмотрим их подробнее.

Windows Forms — это платформа пользовательского интерфейса для создания настольных приложений Windows. Он предоставляет один из самых продуктивных способов создания настольных приложений на основе визуального конструктора, предоставляемого в Visual Studio. Такие функции, как размещение визуальных элементов управления с помощью перетаскивания, упрощают создание настольных приложений.

С помощью Windows Forms вы разрабатываете графически насыщенные приложения, которые легко развертывать, обновлять и работать в автономном режиме или при подключении к Интернету. Приложения Windows Forms могут получить доступ к локальному оборудованию и файловой системе компьютера, на котором выполняется приложение. [C#](https://shwanoff.ru/plus-minus-c-sharp/) является одним из наиболее распространённых языков программирования, который позволяет разрабатывать различные виды приложений на платформах .NET Framework и Windows. C# отличается очень богатым синтаксисом, который в то же время является простым и удобным. Одним из основных достоинств данного языка программирования является наличие мощных функций, и предоставление прямого доступа к операционной памяти.

3.2 Пример работы программного обеспечения

Рассмотрим работу созданного программного продукта. Начальная страница приложения представлена на рисунке 1.

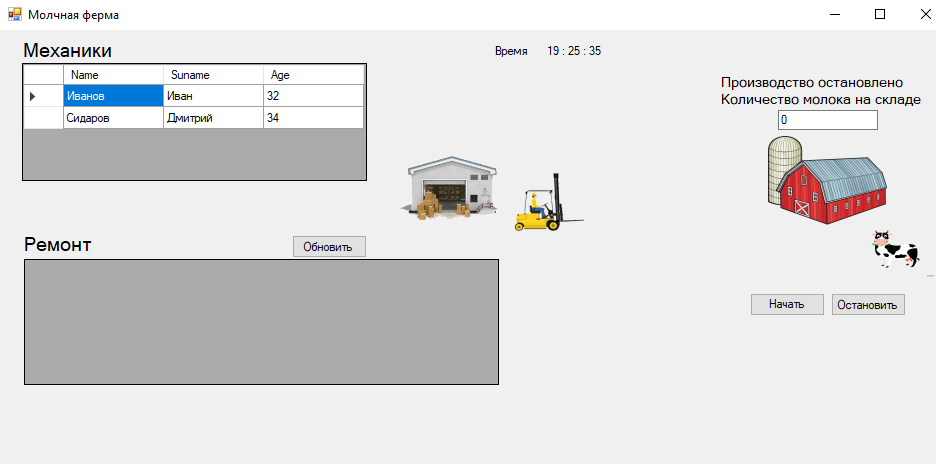


Рисунок 1 – Начальная страница приложения

В данном окне есть список механиков, таймер показывающий текущее время, картину амбара и метку показывающую его состояние и список с ремонтными работами.

Для начала работы молочной фермы нужно нажать кнопку начать, а для того чтобы ее приостановить кнопку остановить (рисунок 2).

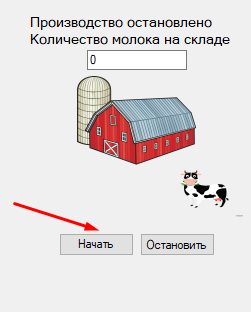


Рисунок 2 – Кнопки запуска, остановки работы

Начнем работу. Можно увидеть как запас молока начал увеличиваться и статус фермы изменился на “Производство работает” (рисунок 3).

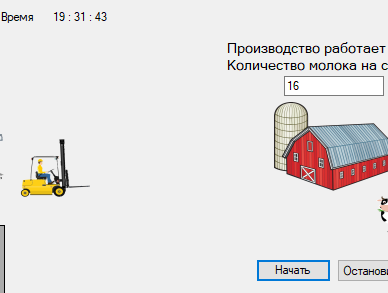


Рисунок 3 – Начало работы

Иногда могут совершатся поломки (рисунок 4).

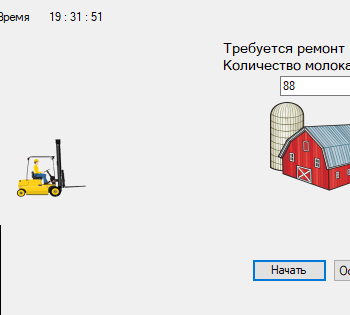


Рисунок 4 – Поломка на производстве

Может произойти ситуация, где амбар заполняется на все сто, тогда приезжает погрузчик и увозит все молоко (рисунок 5,6).



Рисунок 5 – Погрузчик начал движения

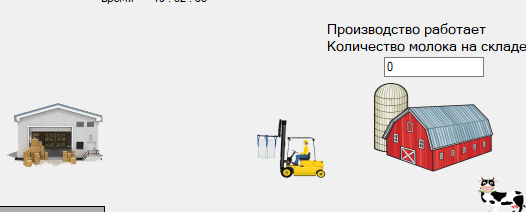


Рисунок 6 – Погрузчик забрал молоко

Можно обновить список ремонтных работ нажав на кнопку “Обновить” (рисунок 7)

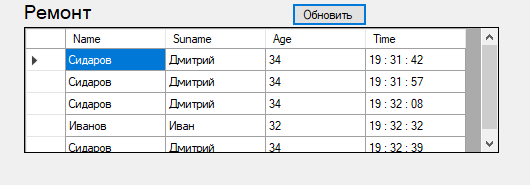


Рисунок 6 – Список ремонтных работ

Таким образом, были рассмотрены основные возможности приложения был разобран функционал, представленный на каждой странице. Производимые команды и действия с данными были проиллюстрированы на примерах. Были описаны основные операции, которые можно производить с данными, отображенными в разработанном приложении. Листинг программного обеспечения приведен в приложении.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были закреплены знания, полученные в процессе изучения дисциплины. Были приобретены навыки самостоятельной разработки программного обеспечения, которые могут быть использованы в разных предметных областях.

Была реализована основные задачи курсового проекта – смоделирована работа молочной фермы, реализованы классы – Механик, Склад, интерфейс – погрузчик. Созданы события – Склад заполнен (приезжает погрузчик и увозит молоко), сломалось оборудование (с некоторой долей вероятности), механик чинит всё. Задача была реализована на языке программирования высокого уровня C#. Программа разработана в среде Microsoft Visual Studio 2021. В ходе выполнения задания была изучена предметная область на тему «Молочная ферма», Созданное приложение было продумано, создан необходимые дополнительные интерфейсы и классы.

Список используемых источников.

1. Орлов С.А. Технологии разработки программного обеспечения: учебное пособие/ Орлов С.А. – М.: Диалог – Наука, 2003. –477 с.
2. Пирогов, В. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В. Пирогов. - СПб.: BHV, 2009. - 528 c.
3. XAML. Введение в язык XAML. [Электронный ресурс]/ Режим доступа: <https://metanit.com/sharp/wpf/2.php>
4. EDM (модель данных с использованием сущностей). [Электронный ресурс]/ Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/entity-data-model>
5. Потоки. [Программирование](https://habr.com/ru/hub/programming/) [C++](https://habr.com/ru/hub/cpp/) [Разработка игр](https://habr.com/ru/hub/gamedev/) [Параллельное программирование/ ). [Электронный ресурс]/](https://habr.com/ru/hub/parallel_programming/)  Режим доступа: <https://habr.com/ru/post/279653/>
6. Чтение и запись текстовых файлов. StreamReader и StreamWriter [Электронный ресурс]/ Режим доступа: https://metanit.com/sharp/tutorial/5.5.php

Приложение

Листинг программного обеспечения

using MilkFactoryLibrary1;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KursovoyGrey

{

public partial class MilkFactForm : Form

{

Loader loader;

Warehouse warehouse ;

public delegate void DelegateForTime(Label label);

DelegateForTime DelTime;

Thread t1;

public MilkFactForm()

{

InitializeComponent();

DelTime = new DelegateForTime(StartTime);

loader = new Loader(pbPogr);

warehouse = new Warehouse(loader,textBox1,lbsost);

}

private void MilkFactForm\_Load(object sender, EventArgs e)

{

t1 = new Thread(LabelTime); // создаем поток

t1.IsBackground = true; // задаем фоновый режым

t1.Priority = ThreadPriority.Lowest; // указываем свмый низкий приоритет

t1.Start(); //

warehouse.mechs.Add(new Mechanic("Иванов", "Иван", 32));

warehouse.mechs.Add(new Mechanic("Сидаров", "Дмитрий", 34));

dataGridView1.DataSource = warehouse.mechs;

}

void LabelTime()

{

// безконечный цыкл

while (true)

{

Invoke(DelTime, lbTime);// запускаем метод с главного потока

}

}

void StartTime(Label label)

{

// выводим всегда две цыфры

// (00:00)

string s = DateTime.Now.Hour.ToString("00");

s += " : ";

s += DateTime.Now.Minute.ToString("00");

s += " : " + DateTime.Now.Second.ToString("00");

label.Text = s;

}

private void ClStr(object sender, EventArgs e)

{

lbsost.Text = "Производство работает";

warehouse.Open();

}

private void ClStp(object sender, EventArgs e)

{

lbsost.Text = "Производство остановлено";

warehouse.Close();

//dataGridView2.DataSource = warehouse.ot;

}

private void ClUpd(object sender, EventArgs e)

{

dataGridView2.DataSource = null;

dataGridView2.DataSource = warehouse.otchet;

}

private void MilkFactForm\_FormClosed(object sender, FormClosedEventArgs e)

{

if(warehouse.IsWork())

warehouse.Close();

t1.Abort();

}

}

}

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MilkFactoryLibrary1

{

public class Loader: ILoader

{

Thread myThread;

PictureBox pb;

static int step = 2;

bool gruzit = false;

public Loader(PictureBox box)

{

pb = box;

}

public void Gruzit()

{

if (gruzit)

{

pb.Image = Image.FromFile("Img//Погр.png");

gruzit = false;

}

else

{

pb.Image = Image.FromFile("Img//ПогрГр.png");

gruzit = true;

}

}

public void ReturnOutAll()

{

myThread = new Thread(new ParameterizedThreadStart(Go));

myThread.Start(this);

}

public static void Go(object obj)

{

//bool gruz = true;

Loader loader = (Loader)obj;

PictureBox pb = loader.pb;

if (pb.Location.X >= 670)

{

Thread.Sleep(1500);

loader.Gruzit();

Warehouse.tbmilk.Invoke((MethodInvoker)delegate

{

Warehouse.tbmilk.Text = "0";

});

Warehouse.lbstatus.Invoke((MethodInvoker)delegate

{

Warehouse.lbstatus.Text = "Производство работает";

});

step = -2;

}

pb.Invoke((MethodInvoker)delegate

{

// Running on the UI thread

pb.Location = new Point(pb.Location.X + step, pb.Location.Y);

});

if (pb.Location.X < 500)

{

loader.Gruzit();

step = 2;

pb.Invoke((MethodInvoker)delegate

{

pb.Location = new Point(pb.Location.X + step, pb.Location.Y);

});

Thread.CurrentThread.Abort();

}

Thread.Sleep(20);

Go(loader);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MilkFactoryLibrary1

{

public class OthetMechanic

{

Mechanic Mech;

DateTime dt;

public OthetMechanic(Mechanic m,DateTime tn) {

Mech = m;

dt = tn;

}

public string Name

{

get

{

return Mech.Name;

}

set

{

Mech.Name = value;

}

}

public string Suname

{

get

{

return Mech.Suname;

}

set

{

Mech.Suname = value;

}

}

public byte Age

{

get

{

return Mech.Age;

}

set

{

Mech.Age = value;

}

}

public string Time

{

get

{

// выводим всегда две цыфры

// (00:00)

string s = dt.Hour.ToString("00");

s += " : ";

s += dt.Minute.ToString("00");

s += " : " + dt.Second.ToString("00");

return s;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MilkFactoryLibrary1

{

public class Mechanic

{

string name = "Default";

string suname = "Default";

byte age = 0;

public Thread myThread;

static object locker = new object();

public Mechanic(string n, string sn, byte a)

{

name = n;

suname = sn;

age = a;

}

public void posEror(List<OthetMechanic> otchet) {

myThread = new Thread(new ParameterizedThreadStart(Work));

myThread.Start((this, otchet));

}

public static void Work(object obj)

{

Thread.Sleep(4000);

lock (locker)

{

if (Warehouse.r.Next(2) == 0 && Warehouse.breaking)

{

Thread.Sleep(3000);

Warehouse.breaking = false;//

Warehouse.lbstatus.Invoke((MethodInvoker)delegate

{

Warehouse.lbstatus.Text = "Производство работает";

});

(Mechanic,List<OthetMechanic>) mechIOtch = ((Mechanic,List<OthetMechanic>))obj;

mechIOtch.Item2.Add(new OthetMechanic(mechIOtch.Item1,DateTime.Now));

}

}

Work(obj);

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Suname

{

get

{

return suname;

}

set

{

suname = value;

}

}

public byte Age

{

get

{

return age;

}

set

{

age = value;

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MilkFactoryLibrary1

{

public class Warehouse

{

public static bool breaking = false;

public static bool full = false;

Loader ld;

public static Random r = new Random();

public static TextBox tbmilk;

public static Label lbstatus;

// const int sizeMilk = 100;

DateTime dt;

Thread myThread=new Thread(new ParameterizedThreadStart(Work));

public List<Mechanic> mechs= new List<Mechanic>();

public List<OthetMechanic> otchet = new List<OthetMechanic>();

public Warehouse(Loader l, TextBox tb,Label label) {

lbstatus= label;

ld = l;

tbmilk = tb;

}

public bool IsWork()

{

return myThread.IsAlive;

}

public void Open()

{

myThread = new Thread(new ParameterizedThreadStart(Work));

myThread.Start(ld);

foreach (var item in mechs)

{

item.posEror(otchet);

}

}

public void Close()

{

myThread.Abort();

}

static void Work(object obj)

{

if (breaking) { Thread.Sleep(3000); Work(obj); return; }

if (tbmilk.Text=="100") {

if (lbstatus.Text.Equals("Производство работает")){

Loader ld = (Loader)obj;

lbstatus.Invoke((MethodInvoker)delegate

{

lbstatus.Text = "Амбор полон";

});

ld.ReturnOutAll();

}

Thread.Sleep(3000);

Work(obj);return;

}

tbmilk.Invoke((MethodInvoker)delegate

{

tbmilk.Text = (Convert.ToInt32(tbmilk.Text) + 1).ToString();

});

if (r.Next(40) == 0)

{

breaking = true;

//tbmilk.Text = "0";//"Требуется ремонт";

lbstatus.Invoke((MethodInvoker)delegate

{

lbstatus.Text = "Требуется ремонт";

});

}

Thread.Sleep(100);

Work(obj);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MilkFactoryLibrary1

{

interface ILoader

{

void ReturnOutAll();

void Gruzit();

}

}